Lecture 10:
Intro to Machine Learning

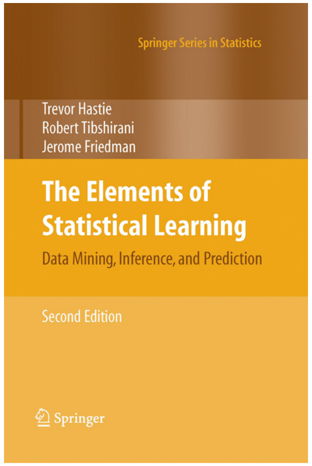
Administrative

HW2 due Friday 2/26

Next ~10 lectures

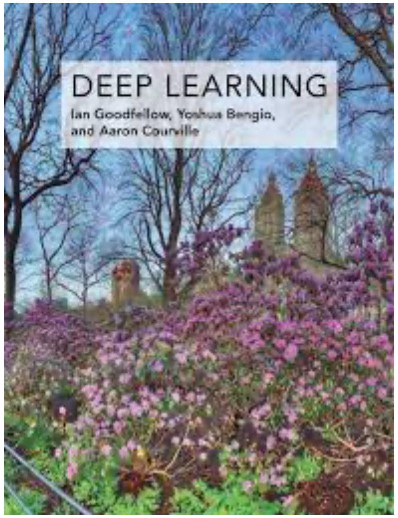
- Machine Learning (ML) + Deep Learning (DL) crash course
- I can't cover everything
- ML really won't solve all problems and is incredibly dangerous if misused
- But ML is a powerful tool and not going away

Pointers



The Elements of Statistical Learning
Hastie, Tibshirani, Friedman
https://web.stanford.edu/~hastie/ElemStatLearn/

Deep Learning
Goodfellow, Bengio, Courville
https://www.deeplearningbook.org/

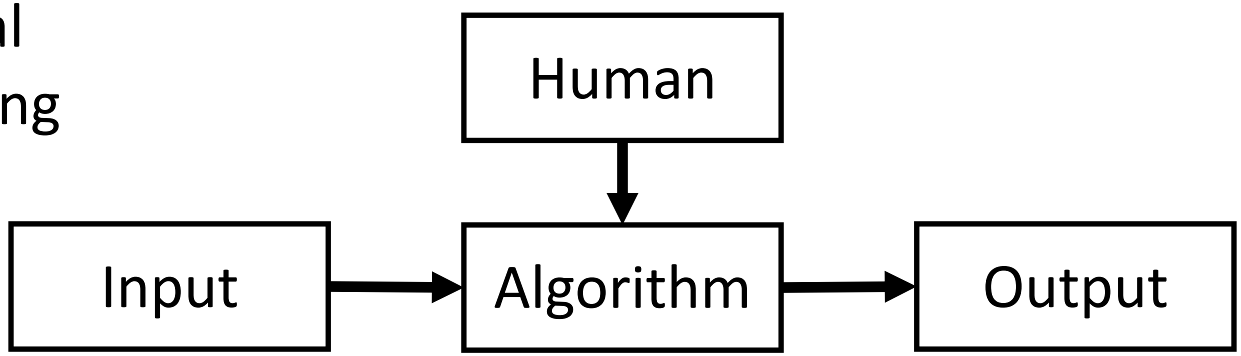


Machine Learning

Algorithms that learn from data

Machine Learning vs Programming

Traditional Programming

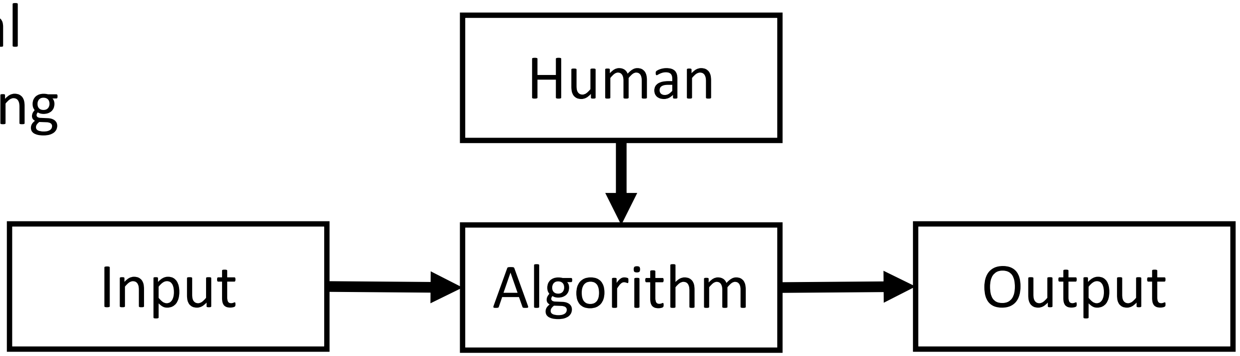


Works well for sorting numbers

```python
def bubble_sort(arr):
  N = len(arr)
  for i in range(N - 1):
    for j in range(N - i - 1):
      if arr[j + 1] < arr[j]:
        temp = arr[j]
        arr[j] = arr[j + 1]
        arr[j + 1] = temp
  return arr
```

Machine Learning vs Programming

Traditional Programming

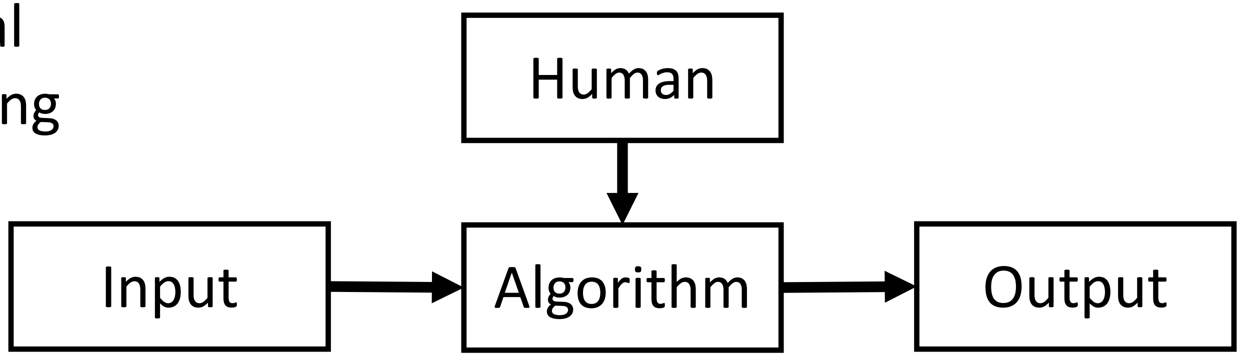


Much harder for some problems

```python
def cat_or_dog(image):
  if ????:
    return "cat"
  else:
    return "dog"
```

Machine Learning vs Programming

Traditional Programming

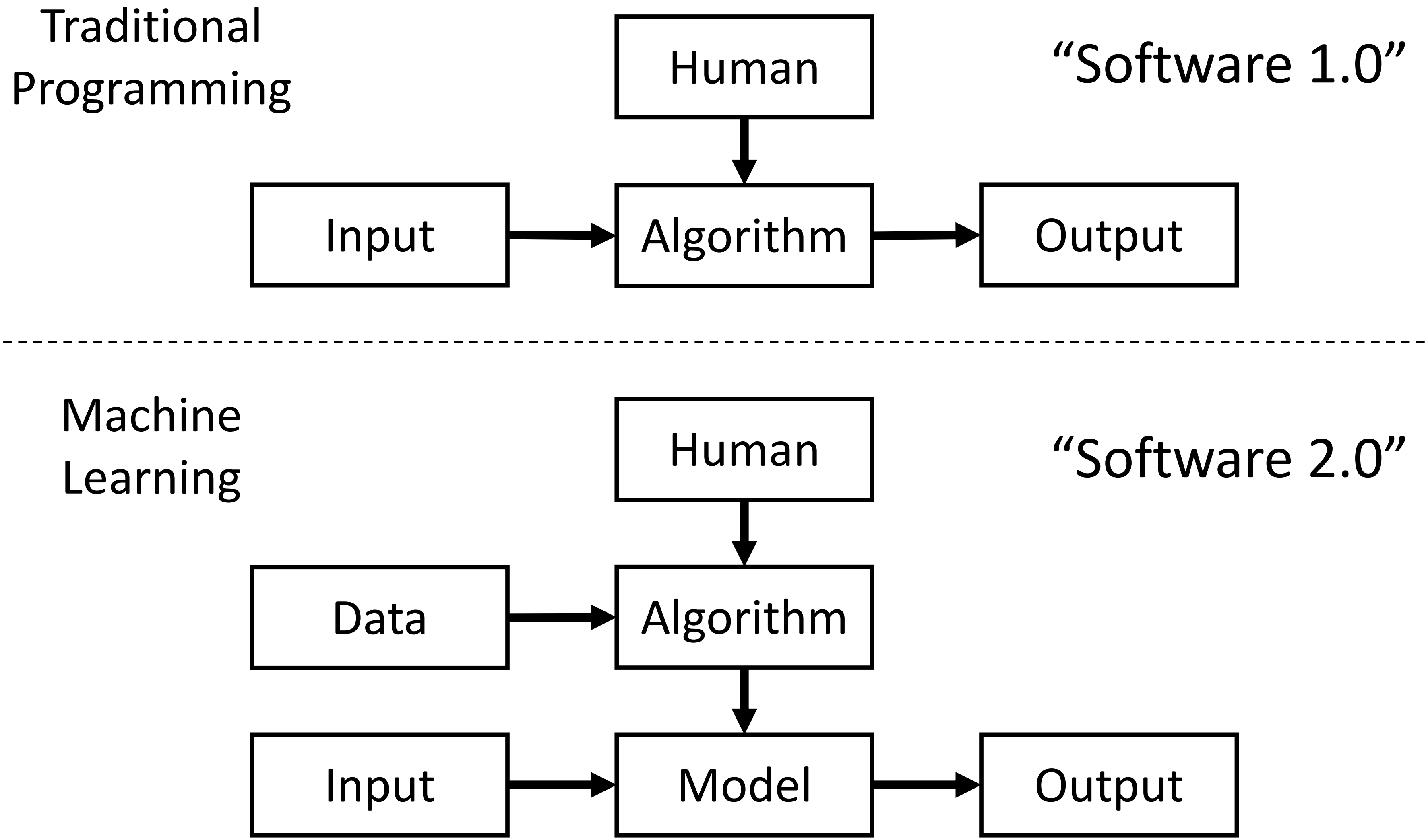


Find edges

Look for ears, whiskers, etc

Problems:
Very brittle
What about car vs truck?

Machine Learning vs Programming



https://medium.com/@karpathy/software-2-0-a64152b37c35

Machine Learning: Data-Driven Approach

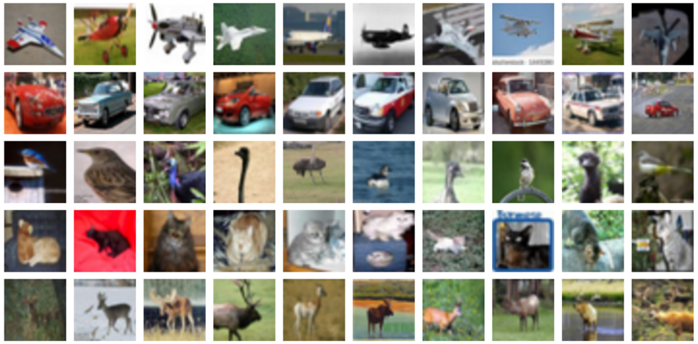
1. Collect a large set of data
2. Use Machine Learning to train a model
3. Evaluate the model on new data

```
def train(images, labels):
  # Machine learning!
  return model
```

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Example training set

airplane
automobile
bird
cat
deer



Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is input / feature

y is label / target

Goal: Learn a *function*
to map x -> y

Supervised vs Unsupervised Learning

Supervised Learning

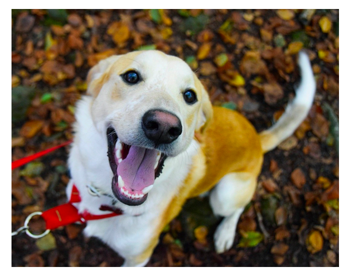
Data: (x, y)

x is input / feature

y is label / target

Goal: Learn a *function* to map x -> y

Image Classification:
Predict a discrete category

x	y
	→ Cat
	→ Dog
	→ Monkey

Supervised vs Unsupervised Learning

Supervised Learning

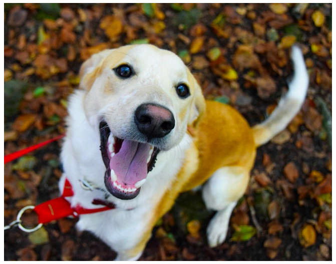
Data: (x, y)

x is input / feature

y is label / target

Goal: Learn a *function* to map x -> y

Image Regression:
Predict a continuous value

x	y
	2 lbs
	25 lbs
	35 lbs

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is input / feature

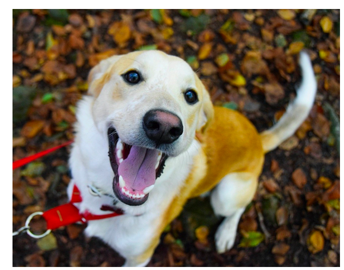
y is label / target

Goal: Learn a *function* to map x -> y

Image Captioning:
Predict a sequence of words

x y

"A white and gray kitten on grass"



"White and orange dog with a red leash in the woods"

"A monkey sitting in front of rocks"

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is input / feature

y is label / target

Goal: Learn a *function* to map x -> y

Unsupervised Learning

Data: x

Just data, no labels!

Goal: Learn underlying *structure* in the data

Supervised vs Unsupervised Learning

Clustering:
Group similar images

Cat image is CC0 public domain
Dog image is CC0 Public Domain
Monkey image is CC0 Public Domain

Cat image is CC0 public domain
Monkey Image is CC0 public domain

Cat image is CC0 public domain
Dog image is CC0 public domain
Dog image is CC0 public domain

Unsupervised Learning

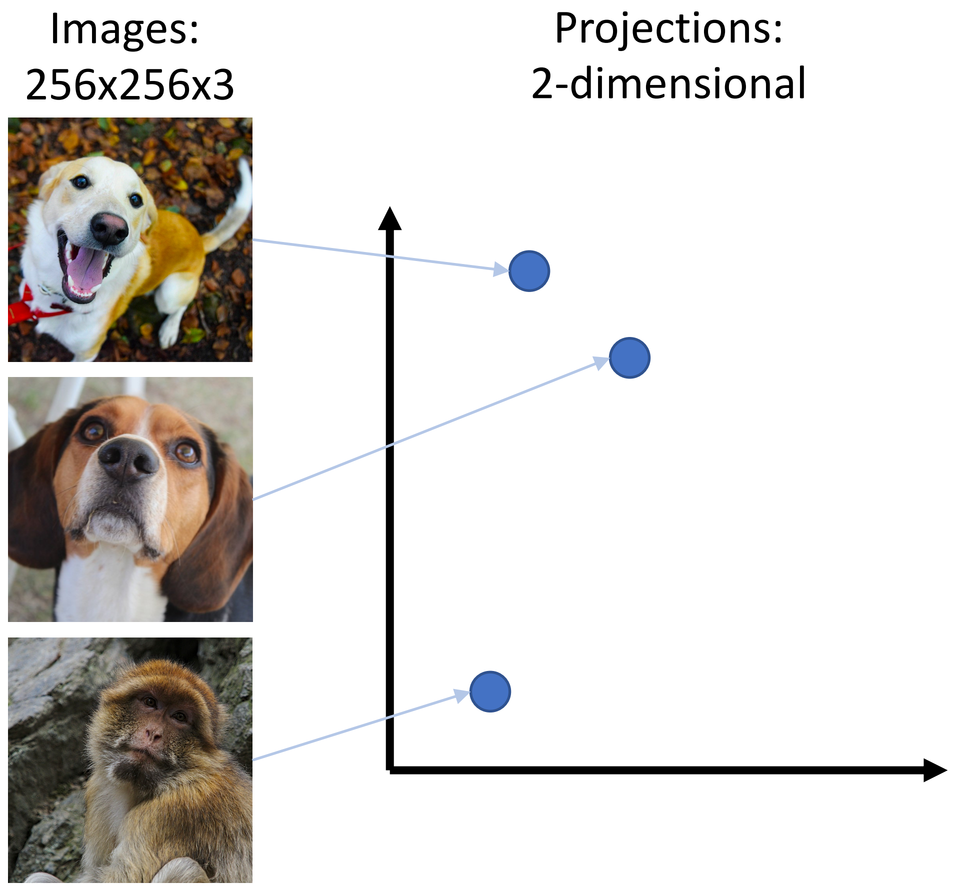
Data: x

Just data, no labels!

Goal: Learn underlying *structure* in the data

Supervised vs Unsupervised Learning

Dimensionality Reduction:
Project to subspace



Unsupervised Learning

Data: x

Just data, no labels!

Goal: Learn underlying *structure* in the data

ML Problems in Vision

	Supervised (Inputs+Labels)	**Unsupervised (Just Data)**
Discrete Output	Classification/ Categorization	Clustering
Continuous Output	Regression	Dimensionality Reduction

Slide adapted from J. Hays

First Machine Learning Algorithm:

Least Squares Linear Regression

Least Squares Linear Regression

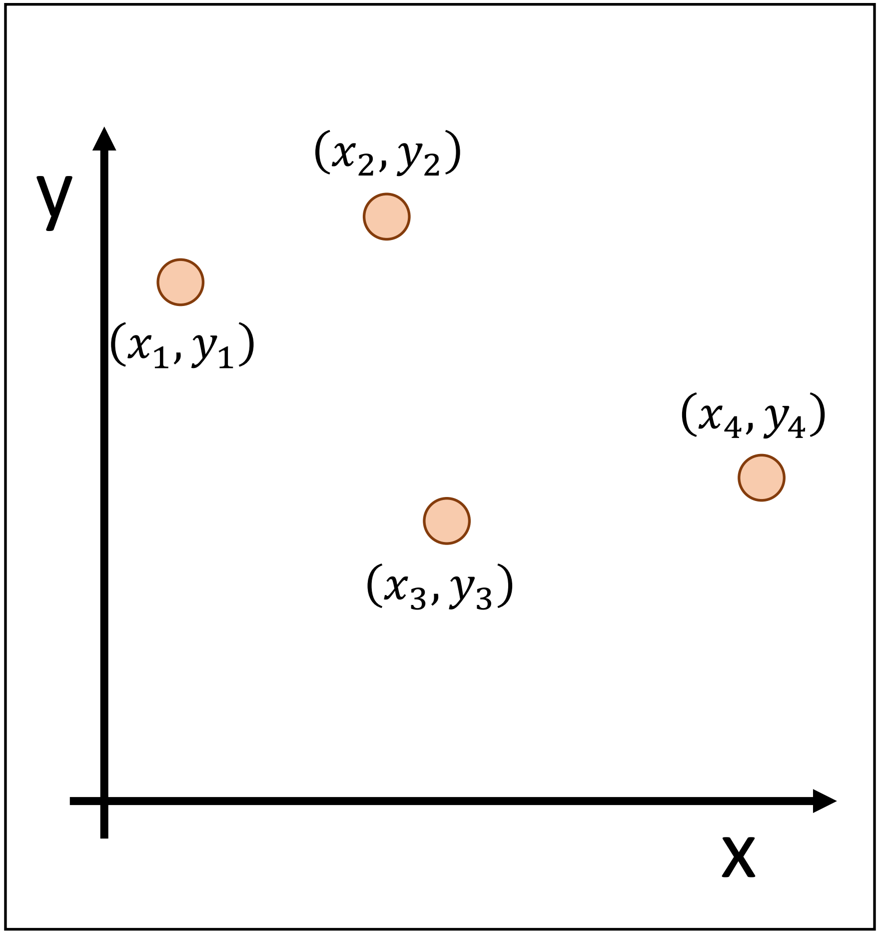
Least Squares Linear Regression

"Regression" = supervised learning with continuous outputs

Data:

$$(x_1, y_1), (x_2, y_2), \dots (x_N, y_N)$$

$$x_i, y_i \in \mathbb{R}$$



Least Squares Linear Regression

"Linear" = Our model is a line

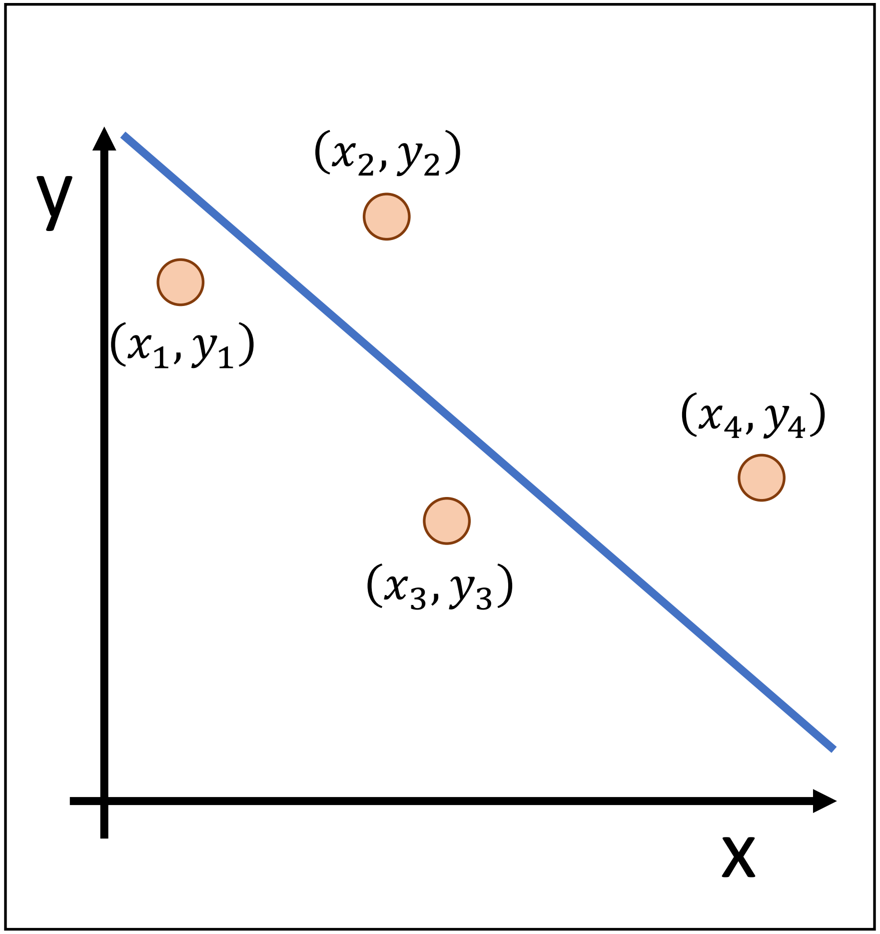
Data:

$$(x_1, y_1), (x_2, y_2), \dots (x_N, y_N)$$
$$x_i, y_i \in \mathbb{R}$$

Model: $y = mx + b$

Or: $\boldsymbol{x} = (x, 1); \boldsymbol{w} = (m, b)$

$y = \boldsymbol{w} \cdot \boldsymbol{x}$



Least Squares Linear Regression

"Least squares" = Find the line that minimizes squared error

Data:

$$(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$$

$$x_i, y_i \in \mathbb{R}$$

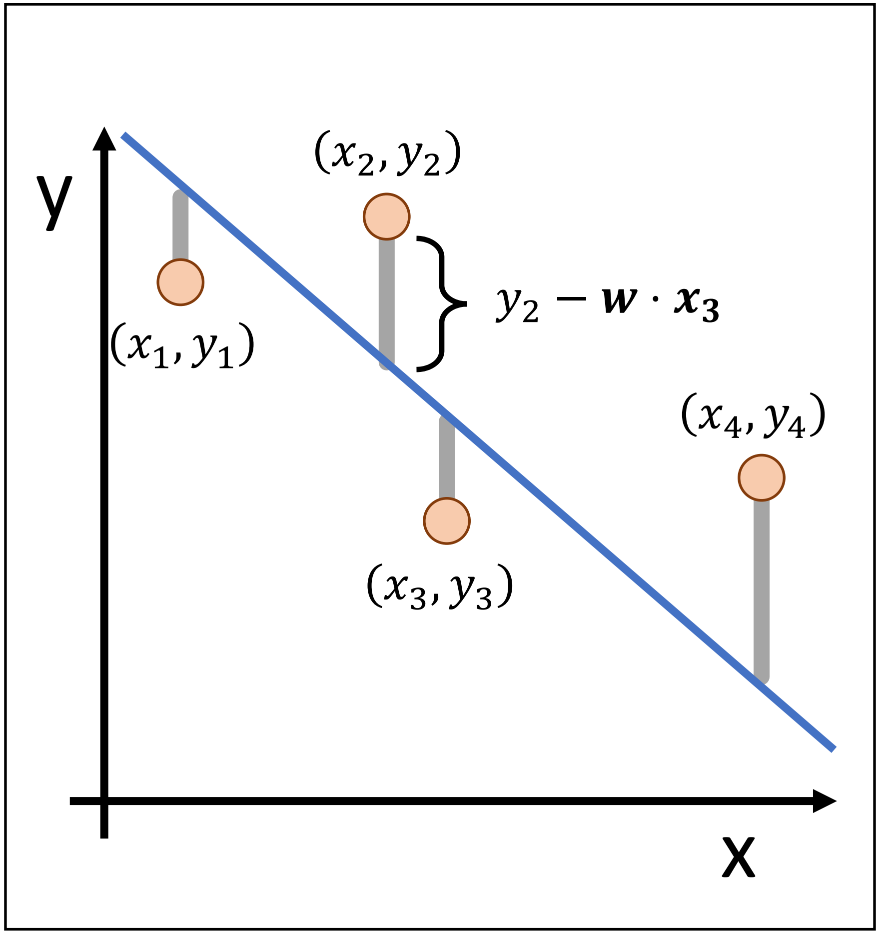
Model: $y = mx + b$

Or: $\boldsymbol{x} = (x, 1); \boldsymbol{w} = (m, b)$

$y = \boldsymbol{w} \cdot \boldsymbol{x}$

Training:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} (y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i)^2$$



Solving Least Squares

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} (y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i)^2$$
$$= \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

Output:
Vector of shape (N,)

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

Inputs:
Matrix of shape (N, 2)

$$\boldsymbol{X} = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix}$$

Weights:
Vector of shape (2,)

$$\boldsymbol{w} = \begin{bmatrix} m \\ b \end{bmatrix}$$

Solution: $\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

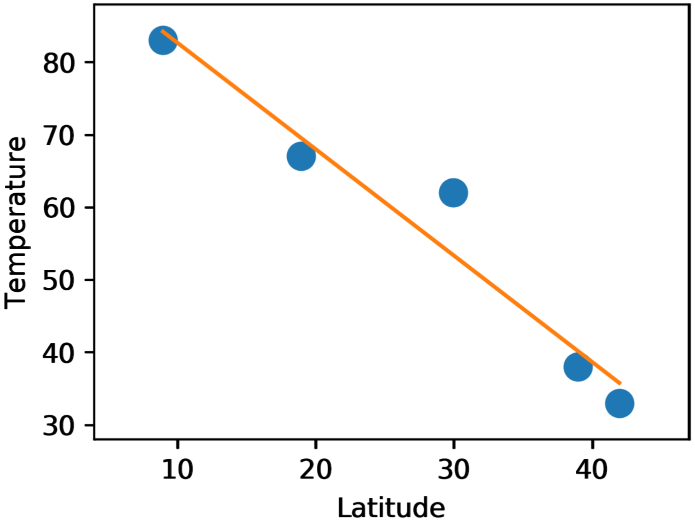
Example: A Bad Weather Model

Given latitude, predict temperature by fitting a line

City	*Latitude (°)*	*Temp (F)*
Ann Arbor	42	33
Washington, DC	39	38
Austin, TX	30	62
Mexico City	19	67
Panama City	9	83

Training

$$\boldsymbol{X}_{5x2} = \begin{bmatrix} 42 & 1 \\ 39 & 1 \\ 30 & 1 \\ 19 & 1 \\ 9 & 1 \end{bmatrix} \quad \boldsymbol{y}_{5x1} = \begin{bmatrix} 33 \\ 38 \\ 62 \\ 67 \\ 83 \end{bmatrix}$$



$$(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

$$w_{2x1} = \begin{bmatrix} -1.47 \\ 97 \end{bmatrix}$$

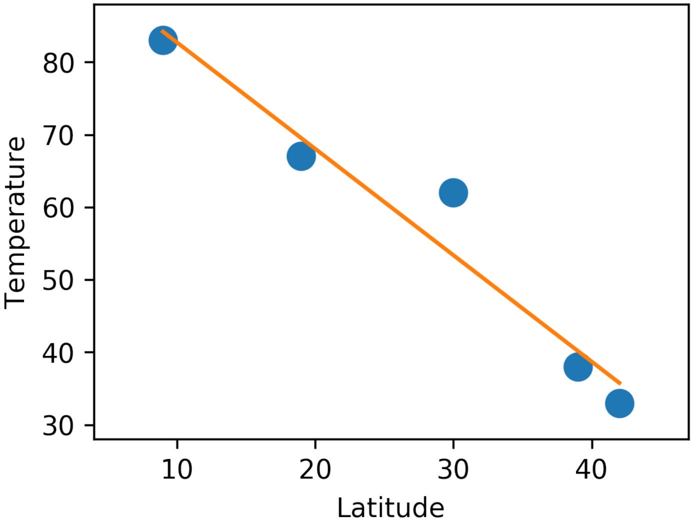
Temp = -1.47*Lat + 97

Example: A Bad Weather Model

Given latitude, predict temperature by fitting a line

City	*Latitude (°)*	*Temp (F)*	*Prediction*	*Error*
Ann Arbor	42	33	35.3	2.3
Washington, DC	39	38	39.7	1.7
Austin, TX	30	62	52.9	10.9
Mexico City	19	67	69.1	2.1
Panama City	9	83	83.8	0.8

Seems good!

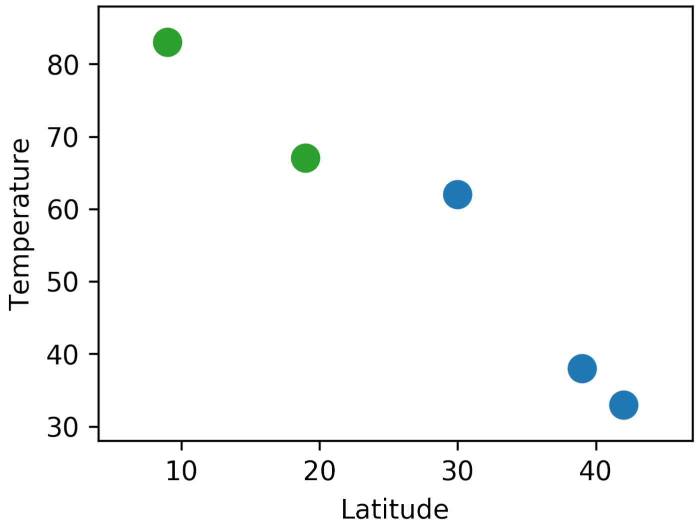


Problem: In ML we don't care about training set performance; we want models that **generalize** to new data

Example: A Bad Weather Model

Given latitude, predict temperature by fitting a line

	City	*Latitude (°)*	*Temp (F)*
Train	Ann Arbor	42	33
	Washington, DC	39	38
	Austin, TX	30	62
Test	Mexico City	19	67
	Panama City	9	83



Problem: In ML we don't care about training set performance; we want models that **generalize** to new data

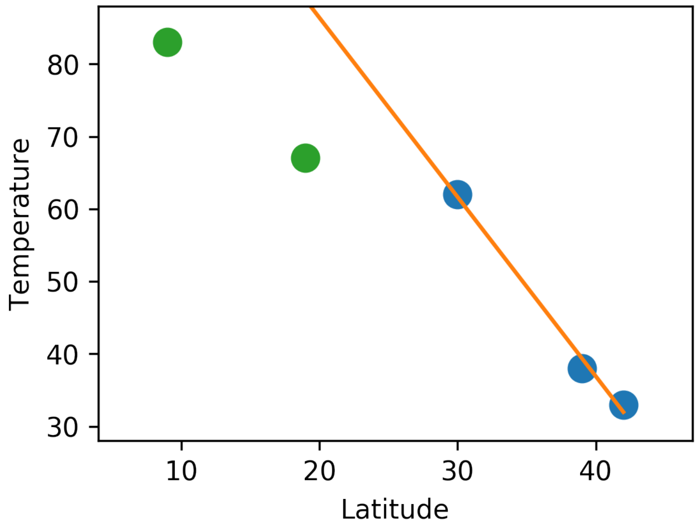
Solution: Split dataset into **train** and **test** sets

Example: A Bad Weather Model

Given latitude, predict temperature by fitting a line

	City	*Latitude (°)*	*Temp (F)*	*Prediction*	*Error*
Train	Ann Arbor	42	33	31.9	1.0
	Washington, DC	39	38	39.4	1.4
	Austin, TX	30	62	61.7	0.3
Test	Mexico City	19	67	88.9	21.9
	Panama City	9	83	113.6	30.6

Problem: Low error on train, high error on test



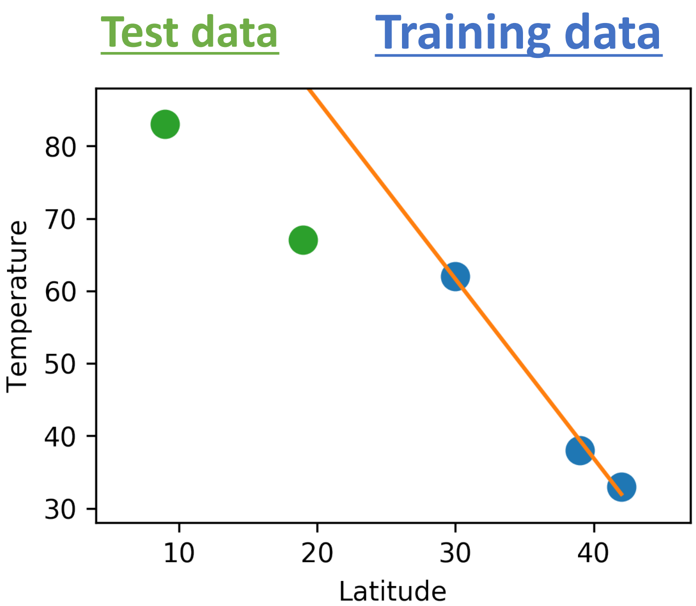
Problem: In ML we don't care about training set performance; we want models that **generalize** to new data

Solution: Split dataset into train and test sets

Fit on train set, evaluate on both

Overfitting

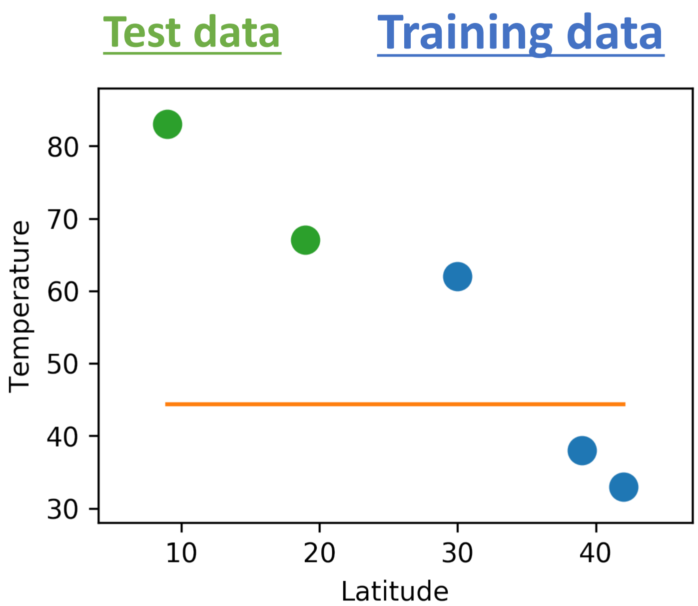
Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is <u>too flexible</u>



Overfitting, Underfitting

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is <u>too flexible</u>

Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is <u>not flexible enough</u>

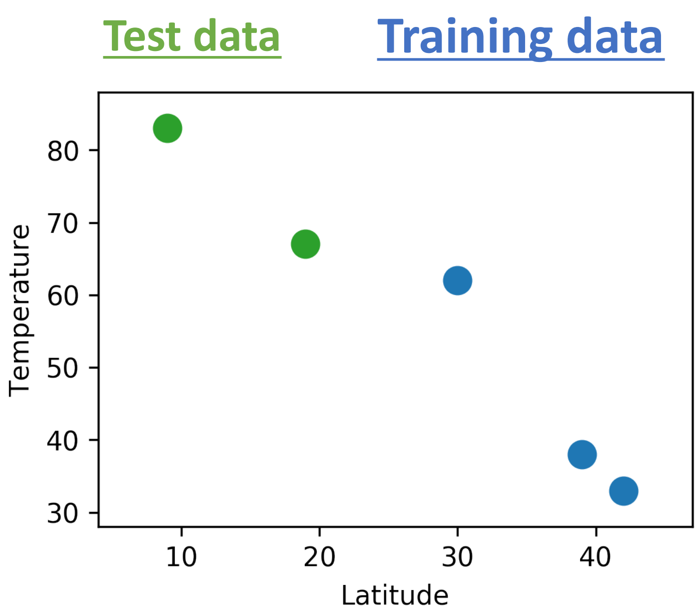


Overfitting, Underfitting, Regularization

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is <u>too flexible</u>

Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is <u>not flexible enough</u>

Regularization: Penalize model complexity to prevent overfitting and improve generalization



Least Squares

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||^2$$

L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||^2 + \lambda||\boldsymbol{w}||^2$$

$$\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Overfitting, Underfitting, Regularization

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is too flexible

Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is not flexible enough

Regularization: Penalize model complexity to prevent overfitting and improve generalization

Least Squares

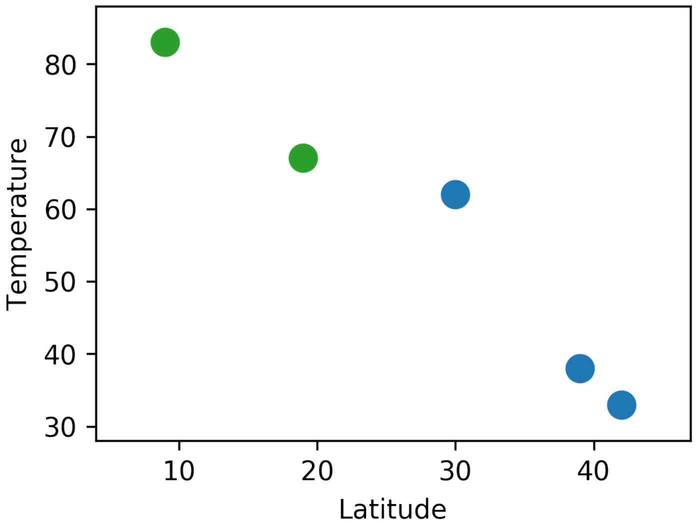
$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

Fit training data

Test data Training data



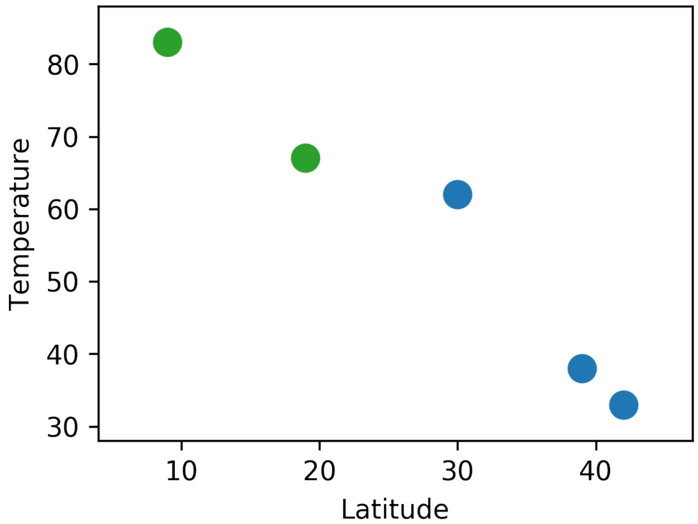
Overfitting, Underfitting, Regularization

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is <u>too flexible</u>

Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is <u>not flexible enough</u>

Regularization: Penalize model complexity to prevent overfitting and improve generalization

<u>Test data</u> <u>Training data</u>



Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda \|\boldsymbol{w}\|^2$$

Fit training data

Penalize complexity

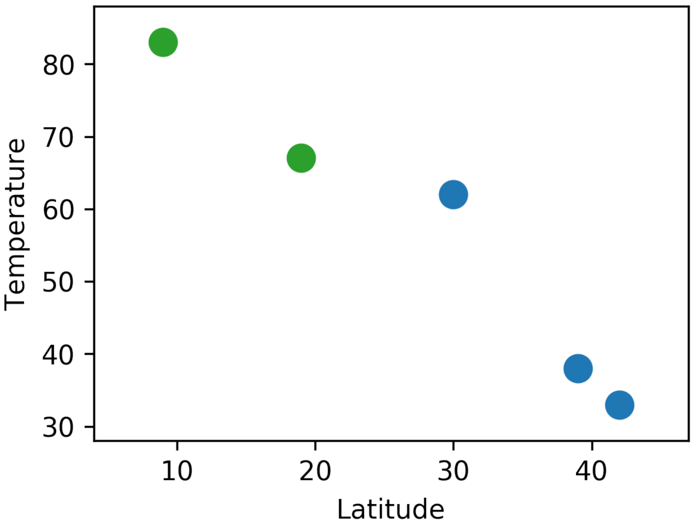
Overfitting, Underfitting, Regularization

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is <u>too flexible</u>

Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is <u>not flexible enough</u>

Regularization: Penalize model complexity to prevent overfitting and improve generalization

Test data   Training data



Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda \|\boldsymbol{w}\|^2$$

Fit training data   Regularization Strength   Penalize complexity

Overfitting, Underfitting, Regularization

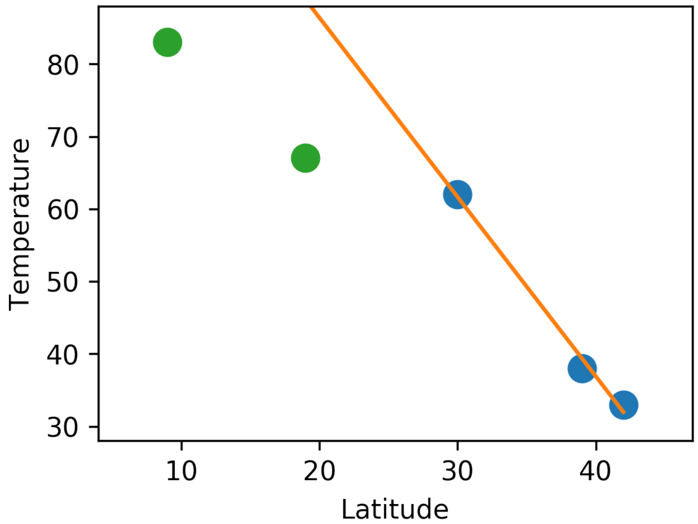
Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is <u>too flexible</u>

Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is <u>not flexible enough</u>

Regularization: Penalize model complexity to prevent overfitting and improve generalization

$\lambda = 0.00$: No regularization; model overfits

<u>Test data</u> <u>Training data</u>



L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

Fit training data **Regularization Strength** **Penalize complexity**

Overfitting, Underfitting, Regularization

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is <u>too flexible</u>

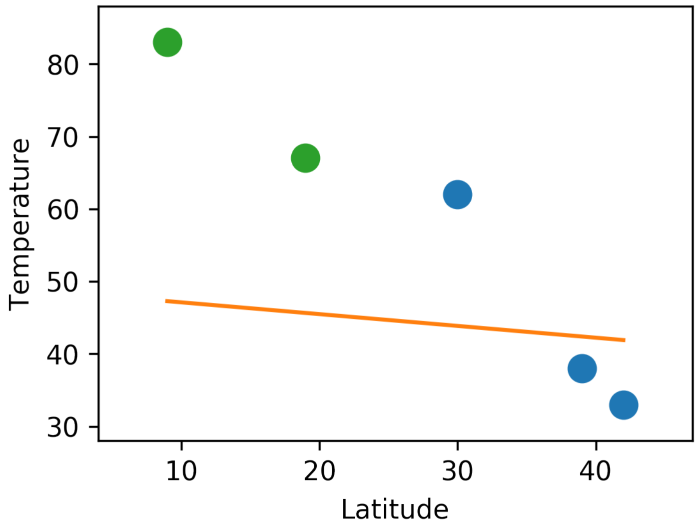
Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is <u>not flexible enough</u>

Regularization: Penalize model complexity to prevent overfitting and improve generalization

$\lambda = 0.00$: No regularization; model overfits

$\lambda = 0.10$: Too much regularization, underfitting

<u>Test data</u> <u>Training data</u>



L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda \|\boldsymbol{w}\|^2$$

Fit training data **Regularization Strength** **Penalize complexity**

Overfitting, Underfitting, Regularization

Overfitting: Model fits noise in the training set and doesn't generalize well to new data
Model is <u>too flexible</u>

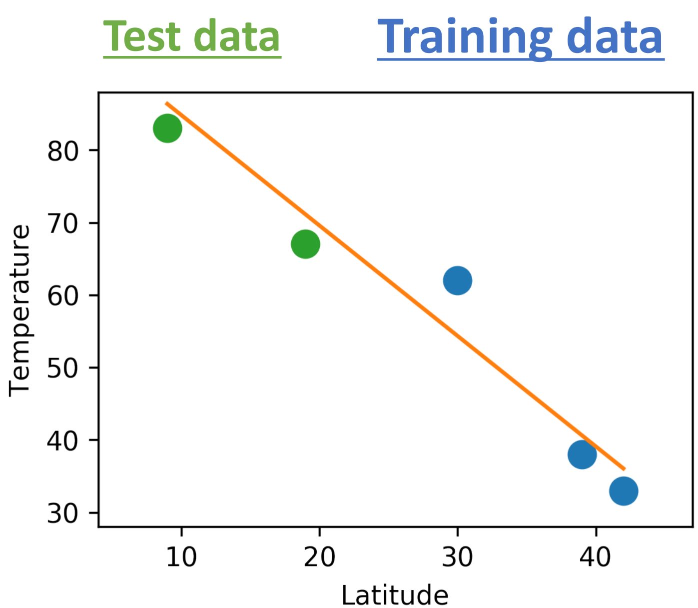
Underfitting: Model doesn't fit the training data well, high error on both train and test
Model is <u>not flexible enough</u>

Regularization: Penalize model complexity to prevent overfitting and improve generalization

$\lambda = 0.00$: No regularization; model overfits

$\lambda = 0.10$: Too much regularization, underfitting

$\lambda = 0.02$: Just right! Good fit and generalization



L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda \|\boldsymbol{w}\|^2$$

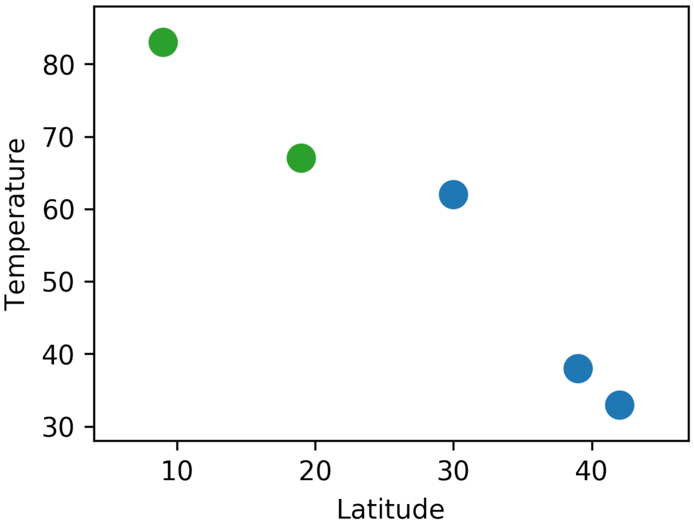
Fit training data

Regularization Strength

Penalize complexity

Regularization

	City	*Latitude (°)*	*Temp (F)*
Train	Ann Arbor	42	33
	Washington, DC	39	38
	Austin, TX	30	62
Test	Mexico City	19	67
	Panama City	9	83



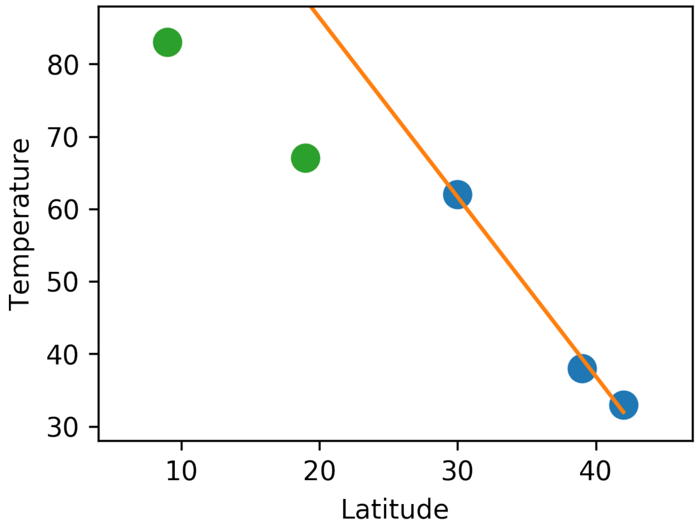
L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda \|\boldsymbol{w}\|^2$$

Fit training data **Regularization Strength** **Penalize complexity**

Regularization

	City	Latitude (°)	Temp (F)	Prediction	Error
				No regularization ($\lambda = 0$)	
Train	Ann Arbor	42	33	31.9	1.0
	Washington, DC	39	38	39.4	1.4
	Austin, TX	30	62	61.7	0.3
Test	Mexico City	19	67	88.9	21.9
	Panama City	9	83	113.6	30.6



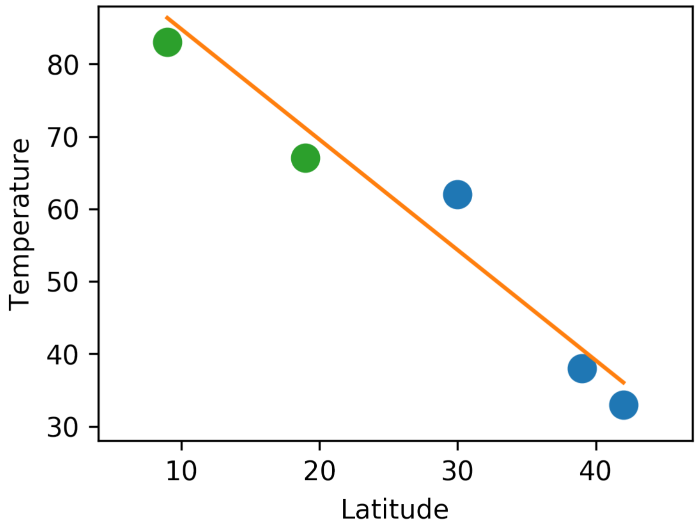
L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \boldsymbol{Xw}||^2 + \lambda ||\boldsymbol{w}||^2$$

Fit training data **Regularization Strength** **Penalize complexity**

Regularization

	City	Latitude (°)	Temp (F)	No regularization ($\lambda = 0$) Prediction	Error	Regularized ($\lambda = 0.02$) Prediction	Error
Train	Ann Arbor	42	33	31.9	1.0	36.0	3.0
	Washington, DC	39	38	39.4	1.4	40.6	2.6
	Austin, TX	30	62	61.7	0.3	54.3	7.7
Test	Mexico City	19	67	88.9	21.9	71.1	4.1
	Panama City	9	83	113.6	30.6	86.4	3.4



L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda \|\boldsymbol{w}\|^2$$

Fit training data **Regularization Strength** **Penalize complexity**

Parameters and Hyperparameters

L2-Regularized Least Squares

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2$$

Parameter ($\boldsymbol{w}$): Selected during training by fitting to training data

Hyperparameter (λ): Chosen before training, does not depend on training data

Question: How to choose hyperparameters?

Choosing Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: λ =0 always works best on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how we will perform on new data

train	test

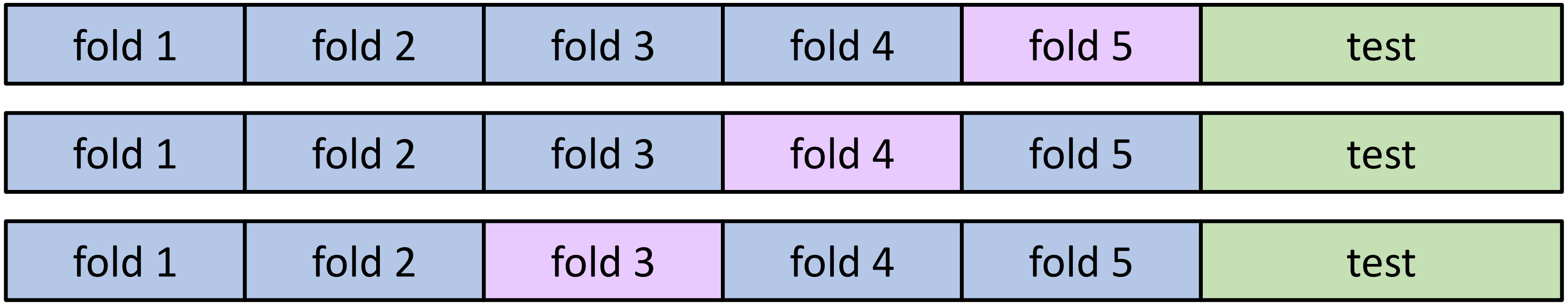
Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

train	validation	test

Choosing Hyperparameters

Idea #4: **Cross-Validation**: Split data into **folds**, try each fold as validation and average the results



Useful for small datasets, but (unfortunately) not used too frequently in deep learning

Least Squares: Multiple Inputs

Instead of scalar inputs $x_i \in \mathbb{R}$,
common to have vector inputs $x_i \in \mathbb{R}^D$

Same equation as before, but terms have different shapes

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} (y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i)^2$$
$$= \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

Output: Vector of shape (N,)

Inputs: Matrix of shape (N, D + 1)

Weights: Vector of shape (D + 1,)

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \quad \boldsymbol{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,D} & 1 \\ \vdots & \ddots & \vdots & 1 \\ x_{N,1} & \cdots & x_{N,D} & 1 \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_D \\ b \end{bmatrix}$$

Solution: $\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

Least Squares: Many Outputs

Vector inputs $x_i \in \mathbb{R}^{D_{in}}$
Vector outputs $y_i \in \mathbb{R}^{D_{out}}$

Same equation as before, but terms have different shapes

$$\boldsymbol{W}^* = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} (y_i - \boldsymbol{W}\boldsymbol{x}_i)^2$$
$$= \arg\min_{\boldsymbol{W}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

Output: Matrix of shape (N, D_{out})

$$\boldsymbol{Y} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,D_{out}} \\ \vdots & \ddots & \vdots \\ y_{N,1} & \cdots & y_{N,D_{out}} \end{bmatrix}$$

Inputs: Matrix of shape (N, D_{in} + 1)

$$\boldsymbol{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,D_{in}} & 1 \\ \vdots & \ddots & \vdots & 1 \\ x_{N,1} & \cdots & x_{N,D_{in}} & 1 \end{bmatrix}$$

Weights: Matrix of shape (D_{in} + 1, D_{out})

$$\boldsymbol{W} = \begin{bmatrix} W_{1,1} & \cdots & W_{D_{out}} \\ \vdots & \ddots & \vdots \\ W_{D_{in},1} & \cdots & W_{D_{in},D_{out}} \\ b_1 & \cdots & b_{D_{out}} \end{bmatrix}$$

Solution: $\boldsymbol{W}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$

Recap

- Machine Learning is a form of data-driven programming
- Supervised Learning maps inputs to outputs
- Unsupervised Learning finds structure in unlabeled data
- ML Example: Least Squares Linear Regression
- Regularization can control overfitting and underfitting
- Use the training set to choose parameters
- Use the validation set to choose hyperparameters
- Use the test set once to estimate generalization

Next Time:
Linear Classifiers